Amendments to the Claims:

This listing of claims replaces all prior versions and listings of claims in the application:

Listing of Claims:

1. (Currently amended) A method of operating a processor comprising:

~~directing~~ executing a branch instruction in execution of an instruction stream with a branch based on ~~any~~ a specified value being true or false and including a first token that specifies the number of instructions in the instruction stream that are after the branch instruction to execute before performing the branch operation and a second token that specifies a branch guess operation.

2. (Currently amended) The method of claim 1 ~~further comprising a~~ wherein the second token that specifies ~~a~~ the branch guess operation if set, pre-fetches a guessed instruction.

3. (currently amended) The method of claim 1 wherein the branch instruction further comprises an optional token that indicates a pipeline stage that the branch operation is evaluated in ~~includes a defer i which causes the processor to execute the ith instruction following the directing before performing the branch operation~~.

4. (Currently amended) The method of claim 1 wherein ~~an optional~~ the first token can specify, one, two or three instructions following to execute before performing the branch operation.

Claim 5 is canceled.

6. (Currently amended) The method of claim 1 wherein one ~~of~~ the ~~optional~~ tokens are specified by a programmer or assembler program to enable variable cycle deferred branching.

7. (Currently amended) The method of claim 1 wherein ~~one of~~ the ~~optional~~ tokens are specified to assist an assembler program to produce more efficient code.

8. (Previously Presented) The method of claim 1 wherein the branch instruction is a branch unconditionally or branch to an instruction at a specified label based on an ALU condition code.

9. (Previously Presented) The method of claim 1 wherein the instruction is a branch to a specific label when a specified bit is set or cleared.

10. (Previously Presented) The method of claim 1 wherein the branch instruction is a branch instruction that causes the processor to branch to the instruction at a specified label if a specified byte in a longword matches or mismatches a byte_compare_value.

11. (Previously Presented) The method of claim 1 wherein the branch instruction is a branch instruction that causes the processor to branch to the instruction at a specified label based on whether or not a current context is a specified context in the branch instruction.

12. (Previously Presented) The method of claim 1 wherein the branch instruction a branch instruction that causes the processor to branch if the state of a specified state name is a selected value.

13. (Previously Presented) The method of claim 1 wherein the branch instruction a branch instruction that causes the processor to branch if a specified signal is deasserted.

14. (Previously Presented) The method of claim 1 wherein the branch instruction further includes an additional token, a guess_branch token which causes the processor to prefetch the instruction for the "branch taken" condition rather than the next sequential instruction.

Claims 15-16. (Canceled)

17. (Previously presented) A method of operating a processor comprises:

executing a branch instruction that causes a branch operation in an instruction stream based on any specified value being true or false; and

deferring performance of the branch operation of the branch instruction based on evaluating a token that specifies a number of instructions to execute before performing the branch operation.

18. (Previously presented) The method of claim 17 further comprising:

evaluating a second token that specifies a branch guess operation, which causes the processor to prefetch an instruction for the "branch taken" condition rather than a next sequential instruction.

19. (Original) The method of claim 17 wherein the optional token is selectable by a programmer.

20. (Original) The method of claim 17 wherein the optional token is specified by a programmer or assembler program to enable variable cycle deferred branching.

21. (Original) The method of claim 17 wherein one of the optional token is specified to assist an assembler program to produce more efficient code.

22. (Previously Presented) A processor comprising:

decode logic for decoding instructions, the decode logic including logic to:

execute a branch instruction in execution of an instruction stream in the processor, with a branch operation based on any specified value in the branch instruction being true or false and

including a token that specifies the number of instructions in the instruction stream to execute before performing the branch operation.

23. (Currently amended) The processor of claim 22 wherein the branch instruction further includes an additional token, a guess_branch token which causes the processor to prefetch the instruction for the "branch taken" condition rather than the next sequential instruction ~~the decode logic comprises logic to execute an ith instruction before performing the branch operation perform based on a value of an optional token in the instruction that a defers execution of the instruction~~.

24. (New) The method of claim 1, wherein the processor is a hardware-based multithreaded processor having multiple engines to process multiple threads and the branch instruction is part of an instruction set for the multiple engines; and executing comprises:

executing the branch on one of the multiple engines.

25. (New) The method of claim 17, wherein the processor is a hardware-based multithreaded processor having multiple engines to process multiple threads and the branch instruction is part of an instruction set for the multiple engines; and executing and deferring occurs on one of the multiple engines.

26. (New) The processor of claim 22, wherein the branch instruction is part of an instruction set for a hardware-based multithreaded processor having multiple engines to process multiple threads.